

DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD  
DDD DDD EEE 888 888 UUU UUU GGG  
DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD

\*\*FILE\*\*ID\*\*DBGNCANCL

G 14

DDDDDDDD	BBBBBBBB	GGGGGGGG	NN	NN	CCCCCCCC	AAAAAA	NN	NN	CCCCCCCC	LL		
DDDDDDDD	BBBBBBBB	GGGGGGGG	NN	NN	CCCCCCCC	AAAAAA	NN	NN	CCCCCCCC	LL		
DD	DD	BB	BB	GG	NN	NN	CC	AA	NN	NN	CC	LL
DD	DD	BB	BB	GG	NN	NN	CC	AA	NN	NN	CC	LL
DD	DD	BB	BB	GG	NNNN	NN	CC	AA	NNNN	NN	CC	LL
DD	DD	BB	BB	GG	NNNN	NN	CC	AA	NNNN	NN	CC	LL
DD	DD	BBBBBBBB	GG	NN	NN	CC	AA	AA	NN	NN	CC	LL
DD	DD	BBBBBBBB	GG	NN	NN	CC	AA	AA	NN	NN	CC	LL
DD	DD	BB	BB	GG	GGGGGG	NN	NNNN	AAAAAAA	NN	NNNN	CC	LL
DD	DD	BB	BB	GG	GGGGGG	NN	NNNN	AAAAAAA	NN	NNNN	CC	LL
DD	DD	BB	BB	GG	GG	NN	NN	AA	AA	NN	CC	LL
DD	DD	BB	BB	GG	NN	NN	CC	AA	AA	NN	CC	LL
DDDDDDDD	BBBBBBBB	GGGGGG	NN	NN	CCCCCCCC	AA	AA	NN	NN	CCCCCCCC	LLLLLLLL	....
DDDDDDDD	BBBBBBBB	GGGGGG	NN	NN	CCCCCCCC	AA	AA	NN	NN	CCCCCCCC	LLLLLLLL	....

LL		SSSSSSS
LL		SSSSSSS
LL		SS
LLLLLLLL		SSSSSSS
LLLLLLLL		SSSSSSS

```

1 0001 0 MODULE DBGNCANCL (IDENT = 'V04-000') =
2 0002 0
3 0003 1 BEGIN
4 0004 1
5 0005 1
6 0006 1 ****
7 0007 1
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1
26 0026 1
27 0027 1 ****
28 0028 1
29 0029 1
30 0030 1 MODULE FUNCTION
31 0031 1 This module contains the command parse and execution networks for
32 0032 1 the CANCEL command.
33 0033 1
34 0034 1 AUTHOR:
35 0035 1 David Plummer
36 0036 1
37 0037 1 CREATION DATE:
38 0038 1 9-JUL-80
39 0039 1
40 0040 1 MODIFIED BY:
41 0041 1 Richard Title 16-Sep-81
42 0042 1
43 0043 1 REVISION HISTORY:
44 0044 1
45 0045 1 3.01 16-SEP-81 RT Implemented CANCEL SOURCE command
46 0046 1 3.02 07-MAY-82 RT Implemented CANCEL DEVELOPER
47 0047 1
48 0048 1
49 0049 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
50 0183 1
51 0184 1 LIBRARY 'LIB$:DBGGEN.L32';
52 0185 1
53 0186 1 FORWARD ROUTINE
54 0187 1 DBGSNPARSE_CANCEL,
55 0188 1 DBGSNEXECUTE_CANCEL; ! ATN parse network for CANCEL
                                ! Command execution network for CANCEL

```

```

57   0189 1 EXTERNAL ROUTINE
58   0190 1   DBG$EVENT_SHOW_CANCEL_SYNTAX,
59   0191 1   DBG$EVENT_SHOW_CANCEL_SEMANTICS,
60   0192 1   DBG$EVENT_CANCEL_ALL,
61   0193 1   DBG$RST_SETSCOPE: NOVALUE,
62   0194 1   DBG$RST_CANMOD,
63   0195 1   DBG$NSAVE_STRING,
64   0196 1   DBG$IS_IT_ENTRY,
65   0197 1   DBG$GET_TEMPMEM,
66   0198 1   DBG$SET_MOD_DEF,
67   0199 1   DBG$NGET_TRANS_RADIX,
68   0200 1   DBG$NMATCH,
69   0201 1   DBG$SCR_EXECUTE_CANDISP_CMD: NOVALUE,
70   0202 1   DBG$SCR_EXECUTE_CANWIND_CMD: NOVALUE,
71   0203 1   DBG$SCR_PARSE_CANDISP_CMD: NOVALUE,
72   0204 1   DBG$SCR_PARSE_CANWIND_CMD: NOVALUE,
73   0205 1   DBG$SRC_CANCEL_SOURCE: NOVALUE,
74   0206 1   DBG$STA_GETSOURCEMOD,
75   0207 1   DBG$SET_STP_DEF: NOVALUE,
76   0208 1   DBG$NSYNTAX_ERROR,
77   0209 1   DBG$NNEXT WORD,
78   0210 1   DBG$NPARSE ADDRESS,
79   0211 1   DBG$NSAVE_DECIMAL_INTEGER,
80   0212 1   DBG$NMAKE_ARG_VECT;
81   0213 1

82   0214 1 EXTERNAL
83   0215 1   DBG$GB_RADIX: VECTOR[3, BYTE],
84   0216 1   DBG$GL_DEVELOPER: BITVECTOR,
85   0217 1   DBG$GL_GBLTYP,
86   0218 1   DBG$GW_GBLNGTH: WORD,
87   0219 1   DBG$GL_DFLTTYP,
88   0220 1   DBG$GW_DFLTLENG: WORD,
89   0221 1   DBG$RUNFRAME: BLOCK [.BYTE],
90   0222 1   DBG$GB_RESIGNAL: BYTE,
91   0223 1   DBG$GL_CONTEXT: BITVECTOR;
92   0224 1

93   0225 1 LITERAL
94   0226 1
95   0227 1   ! Legal verb composites
96   0228 1
97   0229 1   CANCEL_MINIMUM      = 1,
98   0230 1   CANCEL_ALL          = 1,
99   0231 1   CANCEL_BREAK         = 2, ! Also EVENTS_K_CANCEL_BREAK
100  0232 1   CANCEL_BREAK_ALL     = 3,
101  0233 1   CANCEL_EXCEPTION_BREAK = 4, ! Also EVENTS_K_CANCEL_BREAK_EXC
102  0234 1   CANCEL_MODE          = 5,
103  0235 1   CANCEL_MODULE         = 6,
104  0236 1   CANCEL_MODULE_ALL     = 7,
105  0237 1   CANCEL_RADIX          = 20,
106  0238 1   CANCEL_RADIX_OVERRIDE = 21,
107  0239 1   CANCEL_SCOPE          = 8,
108  0240 1   CANCEL_TRACE          = 9, ! Also EVENTS_K_CANCEL_TRACE
109  0241 1   CANCEL_TRACE_CALLS    = 10,
110  0242 1   CANCEL_TRACE_BRANCH    = 11,
111  0243 1   CANCEL_TRACE_ALL      = 12,
112  0244 1   CANCEL_TYPE_OVERRIDE    = 13,
113  0245 1   CANCEL_WATCH          = 14, ! Also EVENTS_K_CANCEL_WATCH

```

J 14  
16-Sep-1984 01:37:15  
14-Sep-1984 12:17:09

VAX-11 Bliss-32 v4.0-742  
[DEBUG.SRC]DBGNCANCL.B32;1

Page 3  
(2)

: 114	0246	1	CANCEL_WATCH_ALL	= 15,
: 115	0247	1	CANCEL_SOURCE	= 16,
: 116	0248	1	CANCEL_DEVELOPER	= 17,
: 117	0249	1	CANCEL_DISPLAY	= 18,
: 118	0250	1	CANCEL_WINDOW	= 19,
: 119	0251	1	CANCEL_MAXIMUM	= 21;

```
: 121      0252 1 GLOBAL ROL .NE DBGSNPARSE_CANCEL (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
: 122      0253 1
: 123      0254 1 ++
: 124      0255 1 FUNCTIONAL DESCRIPTION:
: 125      0256 1
: 126      0257 1 This routine comprises the ATN parse network for the CANCEL verb.
: 127      0258 1 A command execution tree is constructed during the parsing process
: 128      0259 1 which is used as input to the command execution network following
: 129      0260 1 a complete and successful parse. Upon detection of a syntax error,
: 130      0261 1 a message argument vector is constructed and returned.
: 131      0262 1
: 132      0263 1 FORMAL PARAMETERS:
: 133      0264 1
: 134      0265 1 INPUT_DESC      - A longword containing the address of a standard
: 135      0266 1               ASCII string descriptor corresponding to the input
: 136      0267 1               command
: 137      0268 1
: 138      0269 1 VERB_NODE       - A longword containing the address of the command
: 139      0270 1               verb node which is the head node of the command
: 140      0271 1               execution tree
: 141      0272 1
: 142      0273 1 MESSAGE_VECT    - The address of a longword to contain the address of
: 143      0274 1               a standard message argument vector upon detection of
: 144      0275 1               errors
: 145      0276 1
: 146      0277 1 IMPLICIT INPUTS:
: 147      0278 1
: 148      0279 1     NONE
: 149      0280 1
: 150      0281 1 IMPLICIT OUTPUTS:
: 151      0282 1
: 152      0283 1     The command execution tree corresponding to the input command is constructed
: 153      0284 1     on success.
: 154      0285 1
: 155      0286 1     On failure, a message argument vector is constructed and returned.
: 156      0287 1
: 157      0288 1 ROUTINE VALUE:
: 158      0289 1
: 159      0290 1     An unsigned integer longword completion code
: 160      0291 1
: 161      0292 1 COMPLETION CODES:
: 162      0293 1
: 163      0294 1     STSSK_SUCCESS (1)      - Success. Command parsed and execution tree made.
: 164      0295 1
: 165      0296 1     STSSK_SEVERE (4)      - Failure. No tree constructed. Message argument
: 166      0297 1               vector constructed and returned.
: 167      0298 1
: 168      0299 1 SIDE EFFECTS:
: 169      0300 1
: 170      0301 1     NONE
: 171      0302 1
: 172      0303 1
: 173      0304 1
: 174      0305 2 BEGIN
: 175      0306 2
: 176      0307 2 MAP
: 177      0308 2     VERB_NODE: REF DBGSVERB_NODE; ! Pointer to command Verb Node
```

```
178 0309 2
179 0310 2
180 0311 2
181 0312 2
182 0313 2
183 0314 2
184 0315 2
185 0316 2
186 0317 2
187 0318 2
188 0319 2
189 0320 2
190 0321 2
191 0322 2
192 0323 2
193 0324 2
194 0325 2
195 0326 2
196 0327 2
197 0328 2
198 0329 2
199 0330 2
200 0331 2
201 0332 2
202 0333 2
203 0334 2
204 0335 2
205 0336 2
206 0337 2
207 0338 2
208 0339 2
209 0340 2
210 0341 2
211 0342 2
212 0343 2
213 0344 2
214 0345 2
215 0346 2
216 0347 2
217 0348 2
218 0349 2
219 0350 2
220 0351 3
221 0352 3
222 0353 2
223 0354 2
224 0355 2
225 0356 3
226 0357 3
227 0358 3
228 0359 3
229 0360 3
230 0361 3
231 0362 2
232 0363 2
233 0364 2
234 0365 3

: Define strings used at this level of parsing
BIND
DBG$CS_ALL = UPLIT BYTE (%ASCIC 'ALL'),
DBG$CS_BREAK = UPLIT BYTE (%ASCIC 'BREAK'),
DBG$CS_DEVELOPER = UPLIT BYTE (%ASCIC 'DEVELOPÉR'),
DBG$CS_DISPLAY = UPLIT BYTE (%ASCIC 'DISPLAY'),
DBG$CS_EXCEPTION = UPLIT BYTE (%ASCIC 'EXCEPTION'),
DBG$CS_MODE = UPLIT BYTE (%ASCIC 'MODE'),
DBG$CS_MODULE = UPLIT BYTE (%ASCIC 'MODULE'),
DBG$CS_RADIX = UPLIT BYTE (%ASCIC 'RADIX'),
DBG$CS_SCOPE = UPLIT BYTE (%ASCIC 'SCOPE'),
DBG$CS_SOURCE = UPLIT BYTE (%ASCIC 'SOURCE'),
DBG$CS_TRACE = UPLIT BYTE (%ASCIC 'TRACE'),
DBG$CS_TYPE = UPLIT BYTE (%ASCIC 'TYPE'),
DBG$CS_WATCH = UPLIT BYTE (%ASCIC 'WATCH'),
DBG$CS_WINDOW = UPLIT BYTE (%ASCIC 'WINDOW'),
DBG$CS_EQUAL = UPLIT BYTE (%ASCIC '='),
DBG$CS_SLASH = UPLIT BYTE (%ASCIC '/'),
DBG$CS_COMMA = UPLIT BYTE (%ASCIC ','),
DBG$CS_CR = UPLIT BYTE (1, dbg$k_car_return);

LOCAL
STATUS,
NOUN_NODE: REF DBG$NDOUN_NODE; ! Holds routine's return status
! Noun node of command execution tree

! Create and link a noun node. Note that the noun node will not
! be used for certain commands like CANCEL BREAK/ALL.
noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
verb_node [dbg$l_verb_object_ptr] = .noun_node;

! Parse the next keyword and transfer control to a subnetwork
SELECTONE TRUE OF
SET
[dbg$match (.input_desc, dbg$cs_all, 1)] : ! Cancel all
BEGIN
verb_node [dbg$verb_composite] = cancel_all;
END;

[dbg$match (.input_desc, dbg$cs_break, 1)] : ! CANCEL BREAK
BEGIN
VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENTSK_CANCEL_BREAK;
RETURN DBG$EVENT_SHOW_CANCEL_SYNTAX (.INPUT DESC,
.VERB NODE,
.MESSAGE_VECT
);
END;

[dbg$match (.input_desc, dbg$cs_developer, 9)] : ! Set Developer
BEGIN
```

```
: 235      0366 3      LOCAL link;
: 236      0367 3
: 237      0368 3
: 238      0369 3      verb_node [dbg$verb_composite] = cancel_developer;
: 239      0370 3      link = verb_node[dbg$l_verb_object_ptr];
: 240      0371 3      IF NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)
: 241      0372 3      THEN
: 242      0373 4      BEGIN
: 243      0374 4      WHILE true DO
: 244      0375 5      BEGIN
: 245      0376 5      IF NOT dbg$nsave_decimal_integer(.input_desc, noun_node[dbg$l_noun_value],
: 246      0377 5                  .message_vect)
: 247      0378 5      THEN
: 248      0379 5      RETURN sts$sk_severe;
: 249      0380 5
: 250      0381 5      IF (.noun_node[dbg$l_noun_value] LSS 0) OR
: 251      0382 6      (.noun_node[dbg$l_noun_value] GTR 31)
: 252      0383 5      THEN
: 253      0384 6      BEGIN
: 254      0385 6      .message_vect = dbg$nmake_arg_vect(dbg$bitrange);
: 255      0386 6      RETURN sts$sk_severe;
: 256      0387 5      END;
: 257      0388 5
: 258      0389 5      link = noun_node[dbg$l_noun_link];
: 259      0390 5      IF NOT dbg$nmatch(.input_desc, dbg$cs_comma, 1)
: 260      0391 5      THEN
: 261      0392 6      BEGIN
: 262      0393 6      IF NOT dbg$nmatch(.input_desc, dbg$cs_cr, 1)
: 263      0394 6      THEN
: 264      0395 7      BEGIN
: 265      0396 7      .message_vect = dbg$nsyntax_error(dbg$nnext_word(.input_desc));
: 266      0397 7      RETURN sts$sk_severe;
: 267      0398 7      END
: 268      0399 7
: 269      0400 6      ELSE
: 270      0401 6      EXITLOOP;
: 271      0402 6
: 272      0403 5
: 273      0404 5
: 274      0405 5      noun_node = dbg$get_tempmem (dbg$k_noun_node_size);
: 275      0406 5      .link = .noun_node;
: 276      0407 4      END;                                ! End of WHILE loop.
: 277      0408 4
: 278      0409 3
: 279      0410 3
: 280      0411 3
: 281      0412 3
: 282      0413 2
: 283      0414 2
: 284      0415 2
: 285      0416 2      ; Parse the CANCEL DISPLAY command.
: 286      0417 2      [DBG$NMATCH(.INPUT_DESC, DBG$CS_DISPLAY, 3)]:
: 287      BEGIN
: 288      VERB_NODE[DBG$VERB_COMPOSITE] = CANCEL_DISPLAY;
: 289      DBG$SCR_PARSE_CANDISP_CMD(.INPUT_DESC, .VERB_NODE);
: 290      END;
: 291      0422 2
```

```
: 292      0423 2
: 293      0424 2
: 294      0425 2
: 295      0426 2
: 296      0427 2
: 297      0428 3
: 298      0429 3
: 299      0430 3
: 300      0431 3
: 301      0432 3
: 302      0433 3
: 303      0434 4
: 304      0435 4
: 305      0436 5
: 306      0437 5
: 307      0438 5
: 308      0439 5
: 309      0440 5
: 310      0441 5
: 311      0442 4
: 312      0443 4
: 313      0444 3
: 314      0445 3
: 315      0446 3
: 316      0447 3
: 317      0448 3
: 318      0449 3
: 319      0450 3
: 320      0451 3
: 321      0452 3
: 322      0453 2
: 323      0454 2
: 324      0455 2
: 325      0456 3
: 326      0457 3
: 327      0458 2
: 328      0459 2
: 329      0460 2
: 330      0461 3
: 331      0462 3
: 332      0463 3
: 333      0464 3
: 334      0465 3
: 335      0466 3
: 336      0467 4
: 337      0468 4
: 338      0469 4
: 339      0470 4
: 340      0471 4
: 341      0472 4
: 342      0473 5
: 343      0474 5
: 344      0475 6
: 345      0476 6
: 346      0477 6
: 347      0478 6
: 348      0479 5

; Parse the CANCEL EXCEPTION BREAK command.
[dbg$match (.input_desc, dbg$cs_exception, 1)] : ! CANCEL EXCEPTION BREAK
BEGIN

; We look for BREAK
IF NOT dbg$match (.input_desc, dbg$cs_break, 1)
THEN
BEGIN
.message_vect =
(
    IF dbg$match (.input_desc, dbg$cs_cr, 1)
    THEN
        dbg$make_arg_vect (dbg$_needmore)
    ELSE
        dbg$nsyntax_error (dbg$next_word (.input_desc))
);
RETURN sts$severe;
END;

verb_node [dbg$b_verb_composite] = cancel_exception_break;

; Reset the noun and adverb pointers.
verb_node [dbg$l_verb_object_ptr] = 0;
verb_node [dbg$l_verb_adverb_ptr] = 0;
END;

[dbg$match (.input_desc, dbg$cs_mode, 1)] : ! CANCEL MODE
BEGIN
verb_node [dbg$b_verb_composite] = cancel_mode;
END;

[dbg$match (.input_desc, dbg$cs_module, 4)] : ! CANCEL MODULE
BEGIN

; Check for CANCEL MODULE/ALL
IF dbg$match (.input_desc, dbg$cs_slash, 1)
THEN
BEGIN
BIND
    DBG$CS_ALL = UPLIT BYTE (3, 'ALL');

    IF NOT dbg$match (.input_desc, dbg$cs_all, 1)
    THEN
        BEGIN
.message_vect =
(IF dbg$match (.input_desc, dbg$cs_cr, 1)
    THEN
        dbg$make_arg_vect (dbg$_needmore)
    ELSE
        dbg$nsyntax_error (dbg$next_word (.input_desc)));

```

```
349      0480 5          RETURN sts$k_severe;
350      0481 4          END;
351      0482 4
352      0483 4          verb_node [dbg$b_verb_composite] = cancel_module_all;
353      0484 4          END
354
355      0485 4
356      0486 3
357      0487 4
358      0488 4
359      0489 4
360      0490 4
361      0491 4
362      0492 4          BIND    DBG$CS_COMMMA = UPLIT BYTE (1, dbg$k_comma);
363      0493 4          LOCAL   LINK;           ! Temporary pointer
364      0494 4
365      0495 4
366      0496 4
367      0497 4
368      0498 4
369      0499 4
370      0500 5
371      0501 5
372      0502 5
373      0503 5
374      0504 5
375      0505 5
376      0506 5          THEN    RETURN sts$k_severe;
377      0507 5
378      0508 5
379      0509 5
380      0510 5
381      0511 5
382      0512 5          IF NOT dbg$nsmatch (.input_desc, dbg$cs_comma, 1)
383      0513 5          THEN    EXITLOOP;
384
385      0514 5
386      0515 5
387      0516 5
388      0517 5
389      0518 5          link = noun_node [dbg$l_noun_link];
390      0519 5          noun_node = dbg$get_tempmem ?dbg$k_noun_node_size);
391      0520 5          .link = .noun_node;
392      0521 5
393      0522 4          END;       ! End of loop
394
395      0523 4
396      0524 4
397      0525 4
398      0526 4
399      0527 4          noun_node [dbg$l_noun_link] = 0;
400      0528 4
401      0529 4          verb_node [dbg$b_verb_composite] = cancel_module;
402      0530 4
403      0531 3
404      0532 3
405      0533 2
406      0534 2
407      0535 2
408      0536 5          END;
409
410      [dbg$nsmatch (.input_desc, dbg$cs_radix, 1)]:
411      BEGIN
```

```
406      0537 3
407      0538 3
408      0539 3
409      0540 3
410      0541 3
411      0542 3
412      0543 3
413      0544 3
414      0545 3
415      0546 4
416      0547 4
417      0548 4
418      0549 4
419      0550 4
420      0551 4
421      0552 5
422      0553 5
423      0554 6
424      0555 6
425      0556 6
426      0557 6
427      0558 5
428      0559 5
429      0560 4
430      0561 4
431      0562 4
432      0563 3
433      0564 2
434      0565 2
435      0566 2
436      0567 3
437      0568 3
438      0569 2
439      0570 2
440      0571 2
441      0572 3
442      0573 3
443      0574 3
444      0575 3
445      0576 3
446      0577 3
447      0578 3
448      0579 4
449      0580 4
450      0581 4
451      0582 4
452      0583 4
453      0584 4
454      0585 4
455      0586 4
456      0587 4
457      0588 4
458      0589 5
459      0590 5
460      0591 5
461      0592 5
462      0593 4

        BIND
          DBG$CS_OVERRIDE = UPLIT BYTE (8, 'OVERRIDE');
        verb_node[dbg$b_verb_composite] = cancel_radix;
        ! Look for the /
        IF dbg$nmatch (.input_desc, dbg$cs_slash, 1)
        THEN
          BEGIN
            ! Look for 'override'
            IF NOT dbg$nmatch (.input_desc, dbg$cs_override, 1)
            THEN
              BEGIN
                .message_vect =
                  (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
                   THEN
                     dbg$make_arg_vect (dbg$_needmore)
                   ELSE
                     dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
                RETURN sts$k_severe;
              END;
            verb_node [dbg$b_verb_composite] = cancel_radix_override;
          END;
        END;

        [dbg$nmatch (.input_desc, dbg$cs_scope, 1)] :
        BEGIN
          verb_node [dbg$b_verb_composite] = cancel_scope;
        END;

        [dbg$nmatch (.input_desc, dbg$cs_source, 2)] :
        BEGIN
          verb_node[dbg$b_verb_composite] = cancel_source;
          ! Check for CANCEL SOURCE/MODULE=modname
          IF dbg$nmatch (.input_desc, dbg$cs_slash, 1)
          THEN
            BEGIN
              LOCAL
              modnameptr;
              BIND
                dbg$cs_module = UPLIT BYTE (6, 'MODULE');
              IF NOT dbg$nmatch (.input_desc, dbg$cs_module, 4)
              THEN
                BEGIN
                  .message_vect = dbg$nsyntax_error(
                    dbg$nnext_word (.input_desc));
                  RETURN sts$k_severe;
                END;
            END;
```

```
463      0594 4
464      0595 4
465      0596 4
466      0597 4
467      0598 4
468      0599 5
469      0600 5
470      0601 5
471      0602 5
472      0603 4
473      0604 4
474      0605 4
475      0606 4
476      0607 4
477      0608 4
478      0609 4
479      0610 4
480      0611 4
481      0612 4
482      0613 4
483      0614 4
484      0615 4
485      0616 4
486      0617 4
487      0618 4
488      0619 4
489      0620 4
490      0621 4
491      0622 5
492      0623 5
493      0624 5
494      0625 5
495      0626 4
496      0627 4
497      0628 4
498      0629 4
499      0630 3
500      0631 3
501      0632 3
502      0633 3
503      0634 2
504      0635 2
505      0636 2
506      0637 3
507      0638 3
508      0639 3
509      0640 3
510      0641 3
511      0642 3
512      0643 3
513      0644 2
514      0645 2
515      0646 3
516      0647 3
517      0648 3
518      0649 3
519      0650 3

        . Read the = sign
        IF NOT dbg$nmatch (.input_desc, dbg$cs_equal, 1)
        THEN
          BEGIN
            .message_vect = dbg$nsyntax_error(
              dbg$nnext_word(.input_desc));
          RETURN sts$k_severe;
          END;

        ! Read the module name
        IF NOT dbg$nsave_string (.input_desc,
          modnameptr, .message_vect)
        THEN
          RETURN sts$k_severe;

        ! Convert the module name into an rst pointer
        noun_node[dbg$1_noun_value] =
          dbg$sta_getsourcemod(.modnameptr);

        ! If the above routine returns zero then the user has
        ! entered an invalid module name.

        IF .noun_node[dbg$1_noun_value] EQ 0
        THEN
          BEGIN
            .message_vect = dbg$make_arg_vect(
              dbg$nosuchmod, 1, .modnameptr);
          RETURN sts$k_severe;
          END;

        END ! CANCEL SOURCE/MODULE=modname

      ELSE ! the user has just entered CANCEL SOURCE
        noun_node[dbg$1_noun_value] = 0;
      END; ! CANCEL SOURCE

      [dbg$nmatch (.input_desc, dbg$cs_trace, 1)] : ! CANCEL TRACE
      BEGIN
        VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENTSK_CANCEL_TRACE;
        RETURN DBG$EVENT_SHOW_CANCEL_SYNTAX (.INPUT DESC,
          .VERB NODE,
          .MESSAGE_VECT
        );
      END;

      [dbg$nmatch (.input_desc, dbg$cs_type, 2)] : ! CANCEL TYPE/OVERRIDE
      BEGIN
        BIND
          DBG$CS_OVERRIDE = UPLIT BYTE (8, 'OVERRIDE');
      ! Look for the /
    
```

```
520      0651 3
521      0652 3
522      0653 3
523      0654 4
524      0655 4
525      0656 5
526      0657 5
527      0658 5
528      0659 5
529      0660 4
530      0661 4
531      0662 3
532      0663 3
533      0664 3
534      0665 3
535      0666 3
536      0667 3
537      0668 3
538      0669 4
539      0670 4
540      0671 5
541      0672 5
542      0673 5
543      0674 5
544      0675 4
545      0676 4
546      0677 3
547      0678 3
548      0679 3
549      0680 2
550      0681 2
551      0682 2
552      0683 3
553      0684 3
554      0685 3
555      0686 3
556      0687 3
557      0688 3
558      0689 2
559      0690 2
560      0691 2
561      0692 2
562      0693 2
563      0694 3
564      0695 3
565      0696 3
566      0697 2
567      0698 2
568      0699 2
569      0700 2
570      0701 2
571      0702 2
572      0703 3
573      0704 3
574      0705 4
575      0706 4
576      0707 4

      ! IF NOT dbg$nmatch (.input_desc, dbg$cs_slash, 1)
      THEN
        BEGIN
          .message_vect =
            (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
            THEN
              dbg$nmake_arg_vect (dbg$_needmore)
            ELSE
              dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
        RETURN sts$k_severe;
      END;

      ! Look for 'override'
      IF NOT dbg$nmatch (.input_desc, dbg$cs_override, 1)
      THEN
        BEGIN
          .message_vect =
            (IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
            THEN
              dbg$nmake_arg_vect (dbg$_needmore)
            ELSE
              dbg$nsyntax_error (dbg$nnext_word (.input_desc)));
        RETURN sts$k_severe;
      END;

      verb_node [dbg$b_verb_composite] = cancel_type_override;
      END;

[dbg$nmatch (.input_desc, dbg$cs_watch, 1)] : ! CANCEL WATCH
      BEGIN
        VERB_NODE [DBG$B_VERB_COMPOSITE] = EVENT$K_CANCEL_WATCH;
        RETURN DBG$EVENT_SHOW_CANCEL_SYNTAX (.INPUT_DESC,
                                              .VERB_NODE,
                                              .MESSAGE_VECT
                                             );
      END;

      ! Parse the CANCEL WINDOW command.
[DBG$NMATCH(.INPUT_DESC, DBG$CS_WINDOW, 3)]:
      BEGIN
        VERB_NODE[DBG$B_VERB_COMPOSITE] = CANCEL WINDOW;
        DBG$SCR_PARSE_C$ANWIN$CMD(.INPUT_DESC, .VERB_NODE);
      END;

      ! Any other CANCEL command constitutes a syntax error.
[OTHERWISE] : ! Syntax error
      BEGIN
        .message_vect =
        (
          IF dbg$nmatch (.input_desc, dbg$cs_cr, 1)
          THEN
```

```

577      0708 4      dbg$make_arg_vect (dbg$needmore)
578      0709 4      ELSE
579      0710 4      dbg$syntax_error (dbg$next_word (.input_desc))
580      0711 3      );
581      0712 3      RETURN st$K_severe;
582      0713 2      END;
583      0714 2
584      0715 2      TES:
585      0716 2
586      0717 2      RETURN ST$K_SUCCESS;
587      0718 2
588      0719 1      END;

```

```

.TITLE (DBGNCANCL
.IDENT  \V04-000\
.PSECT  DBG$PLIT,NOWRT, SHR, PIC,0

```

52	45	50	4F	4B	41	4C	4C	41	03	00000 P.AAA:	.ASCII <3>\ALL\	
	59	41	4C	45	56	52	42	05	00004 P.AAB:	.ASCII <5>\BREAK\		
4E	4F	49	54	50	45	43	58	44	09	0000A P.AAC:	.ASCII <9>\DEVELOPER\	
					45	44	4F	4D	07	00014 P.AAD:	.ASCII <7>\DISPLAY\	
					45	4C	55	44	09	0001C P.AAE:	.ASCII <9>\EXCEPTION\	
					58	49	44	41	04	00026 P.AAF:	.ASCII <4>\MODE\	
					45	50	4F	43	06	0002B P.AAG:	.ASCII <6>\MODULE\	
					58	49	44	41	05	00032 P.AAH:	.ASCII <5>\RADIX\	
					45	50	55	4F	05	00038 P.AAI:	.ASCII <5>\SCOPE\	
					45	43	52	53	06	0003E P.AAJ:	.ASCII <6>\SOURCE\	
					45	43	41	52	05	00045 P.AAK:	.ASCII <5>\TRACE\	
					45	45	50	59	04	0004B P.AAL:	.ASCII <4>\TYPE\	
					57	48	43	54	05	00050 P.AAM:	.ASCII <5>\WATCH\	
					57	4F	44	4E	06	00056 P.AAN:	.ASCII <6>\WINDOW\	
						3D	01	0005D P.AAO:	.ASCII <1>\=\			
						2F	01	0005F P.AAP:	.ASCII <1>\/\			
						2C	01	00061 P.AAQ:	.ASCII <1>\\\			
						0D	01	00063 P.AAR:	.BYTE 1, 13			
							03	00065 P.AAS:	.BYTE 3			
						4C	4C	41	00066 P.AAT:	.ASCII \ALL\		
						2C	01	00069 P.AAU:	.BYTE 1, 44			
							08	0006B P.AAV:	.BYTE 8			
						45	44	49	52	56	06 P.AAV:	.ASCII \ OVERRIDE\
						45	4C	55	44	4F	00074 P.AAW:	.BYTE 6
									08	00075 P.AAW:	.ASCII \ MODULE\	
						45	44	49	52	56	4F 0007C P.AAW:	.ASCII \ OVERRIDE\

DBG\$CS_ALL=	P.AAA
DBG\$CS_BREAK=	P.AAB
DBG\$CS_DEVELOPER=	P.AAC
DBG\$CS_DISPLAY=	P.AAD
DBG\$CS_EXCEPTION=	P.AAE
DBG\$CS_MODE=	P.AAF
DBG\$CS_MODULE=	P.AAG
DBG\$CS_RADIX=	P.AAH
DBG\$CS_SCOPE=	P.AAI
DBG\$CS_SOURCE=	P.AAJ

DBGSCS\_TRACE= P.AAK  
 DBGSCS\_TYPE= P.AAL  
 DBGSCS\_WATCH= P.AAM  
 DBGSCS\_WINDOW= P.AAN  
 DBGSCS\_EQUAL= P.AAO  
 DBGSCS\_SLASH= P.AAP  
 DBGSCS\_COMMAS= P.AAQ  
 DBGSCS\_CR= P.AAR  
 DBGSCS\_ALL= P.AAS  
 DBGSCS\_COMMAS= P.AAT  
 DBGSCS\_OVERRIDE= P.AAU  
 DBGSCS\_MODULE= P.AAV  
 DBGSCS\_OVERRIDE= P.AAW  
 .EXTRN DBGS\$EVENT\_SHOW\_CANCEL\_SYNTAX  
 .EXTRN DBGS\$EVENT\_SHOW\_CANCEL\_SEMANTICS  
 .EXTRN DBGS\$RST\_SETSCOPE  
 .EXTRN DBGS\$RST\_CANMOD, DBGS\$NSAVE\_STRING  
 .EXTRN DBGSIS\_IT\_ENTRY  
 .EXTRN DBGS\$GET\_TEMPMM  
 .EXTRN DBGS\$SET\_MOD\_DEF  
 .EXTRN DBGS\$NGET\_TRANS\_RADIX  
 .EXTRN DBGS\$NMATCH, DBGS\$SCR\_EXECUTE\_CANDISP\_CMD  
 .EXTRN DBGS\$SCR\_EXECUTE\_CANWIND\_CMD  
 .EXTRN DBGS\$SCR\_PARSE\_CANDISP\_CMD  
 .EXTRN DBGS\$SCR\_PARSE\_CANWIND\_CMD  
 .EXTRN DBGS\$SRC\_CANCEL\_SOURCE  
 .EXTRN DBGS\$STA\_GETSOURCEMOD  
 .EXTRN DBGS\$SET\_STP\_DEF  
 .EXTRN DBGS\$NSYNTAX\_ERROR  
 .EXTRN DBGS\$NNEXT\_WORD, DBGS\$NPARSE\_ADDRESS  
 .EXTRN DBGS\$NSAVE\_DECIMAL\_INTEGER  
 .EXTRN DBGS\$MAKE\_ARG\_VECT  
 .EXTRN DBGS\$GB\_RADIX, DBGS\$GL\_DEVELOPER  
 .EXTRN DBGS\$GL\_GBLTYP, DBGS\$GL\_GBLNGTH  
 .EXTRN DBGS\$GL\_DFLTTYP, DBGS\$GL\_DFLTLENG  
 .EXTRN DBGS\$RUNFRAME, DBGS\$GB\_RESIGNAL  
 .EXTRN DBGS\$GL\_CONTEXT  
 .PSECT DBGS\$CODE,NOWRT, SHR, PIC,0

		07FC 00000		
			.ENTRY	DBGS\$NPARSE_CANCEL, Save R2,R3,R4,R5,R6,R7,- : 0252
				R8,R9,R10
		5A 00000000G 00 9E 00002	MOVAB	DBGS\$MAKE_ARG_VECT, R10
		59 00000000G 00 9E 00009	MOVAB	DBGS\$NSAVE_STRING, R9
		58 00000000G 00 9E 00010	MOVAB	DBGS\$GET_TEMPMM, R8
		57 00000000G 00 9E 00017	MOVAB	DBGS\$NMATCH, R7
		56 00000000' EF 9E 0001E	MOVAB	DBGSCS_SLASH, R6
		5E 04 C2 00025	SUBL2	#4, SP
		04 DD 00028	PUSHL	#4
		68 01 FB 0002A	CALLS	#1, DBGS\$GET_TEMPMM
		54 50 D0 0002D	MOVL	R0, NOUN_NODE
08	A3	04 AC 7D 00030	MOVQ	INPUT_DESC, R2
		52 54 D0 00034	MOVL	NOUN_NODE, 8(R3)
		01 DD 00038	PUSHL	#1
		A1 A6 9F 0003A	PUSHAB	DBGSCS_ALL
		52 DD 0003D	PUSHL	R2

0341

0350

0342

0350

		67	03	FB 0003F		CALLS #3, DBG\$NMATCH	
		01	50	D1 00042		CMPL R0, #1	
			07	12 00045		BNEQ 1\$	
		01	01	90 00047		MOV B #1 1(R3)	0352
	01	A3	02B7	31 00048	1\$:	BRW 43\$	0347
			01	DD 0004E		PUSHL #1	0355
			A5	A6 9F 00050		PUSHAB DBG\$CS_BREAK	
				52 DD 00053		PUSHL R2	
		67	03	FB 00055		CALLS #3, DBG\$NMATCH	
	01		50	D1 00058		CMPL R0, #1	
			07	12 00058		BNEQ 2\$	
	01	A3	02	90 0005D		MOV B #2 1(R3)	0357
			0244	31 00061		BRW 35\$	0360
			09	DD 00064	2\$:	PUSHL #9	0364
			AB	A6 9F 00066		PUSHAB DBG\$CS_DEVELOPER	
				52 DD 00069		PUSHL R2	
		67	03	FB 0006B		CALLS #3, DBG\$NMATCH	
	01		50	D1 0006E		CMPL R0, #1	
			6B	12 00071		BNEQ 9\$	
	01	A3	11	90 00073		MOV B #17 1(R3)	0369
		55	08	A3 9E 00C77		MOVAB 8(R2), LINK	0370
			01	DD 0007B		PUSHL #1	0371
			04	A6 9F 0007D		PUSHAB DBG\$CS_CR	
			52	DD 00080		PUSHL R2	
		67	03	FB 00082		CALLS #3, DBG\$NMATCH	
	52		50	E8 00085		BLBS R0, 8\$	
			0C	AC DD 00088	3\$:	PUSHL MESSAGE_VECT	0377
				14 BB 0008B		PUSHR #^M<R2,R4>	0376
00000000G	00	03	03	FB 0008D		CALLS #3, DBG\$NSAVE_DECIMAL_INTEGER	
			50	E8 00094		BLBS R0, 4\$	
			0267	31 00097		BRW 42\$	
			64	D5 0009A	4\$:	TSTL (NOUN_NODE)	0381
		1F	05	19 0009C		BLSS 5\$	
			64	D1 0009E		CMPL (NOUN_NODE), #31	0382
			09	15 000A1		BLEQ 6\$	
		00028248	8F	DD 000A3	5\$:	PUSHL #164424	0385
			023A	31 000A9		BRW 39\$	
	55		08	A4 9E 000AC	6\$:	MOVAB 8(R4), LINK	0389
			01	DD 000B0		PUSHL #1	0390
			02	A6 9F 000B2		PUSHAB DBG\$CS_COMMA	
			52	DD 000B5		PUSHL R2	
		67	03	FB 000B7		CALLS #3, DBG\$NMATCH	
	10		50	E8 000BA		BLBS R0, 7\$	
			01	DD 000BD		PUSHL #1	0393
			04	A6 9F 000BF		PUSHAB DBG\$CS_CR	
			52	DD 000C2		PUSHL R2	
		67	03	FB 000C4		CALLS #3, DBG\$NMATCH	
	10		50	E8 000C7		BLBS R0, 8\$	
			021E	31 000CA		BPW 40\$	0396
			04	DD 000CD	7\$:	PUSHL #4	0405
	68		01	FB 000CF		CALLS #1, DBG\$GET_TEMP MEM	
			50	DD 000D2		MOVL R0, NOUN_NODE	
	54		54	DD 000D5		MOVL NOUN_NODE, (LINK)	0406
			AE	11 000D8		BRB 3\$	0374
			65	D4 000DA	8\$:	CLRL (LINK)	0411
			56	11 000DC		BRB 12\$	0347
			03	DD 000DE	9\$:	PUSHL #3	0418

		B5	A6	9F 000E0	PUSHAB	DBG\$CS_DISPLAY	
		52	DD 000E3	PUSHL	R2		
		03	FB 000E5	CALLS	#3, DBGSNMATCH		
		50	D1 000E8	CMPL	R0, #1		
		0F	12 000EB	BNEQ	10\$		
		12	90 000ED	MOV	#18, 1(R3)	0420	
01	A3	OC	BB 000F1	PUSHR	#^M<R2,R3>	0421	
0000000G 00		02	FB 000F3	CALLS	#2, DBG\$SCR_PARSE_CANDISP_CMD	0347	
		66	11 000FA	BRB	15\$	0427	
		01	DD 000FC	PUSHL	#1		
		BD	A6 9F 000FE	PUSHAB	DBG\$CS_EXCEPTION		
		52	DD 00101	PUSHL	R2		
		03	FB 00103	CALLS	#3, DBGSNMATCH		
		50	D1 00106	CMPL	R0, #1		
		16	12 00109	BNEQ	11\$		
		01	DD 0010B	PUSHL	#1		
		A5	A6 9F 0010D	PUSHAB	DBG\$CS_BREAK	0432	
		52	DD 00110	PUSHL	R2		
		03	FB 00112	CALLS	#3, DBGSNMATCH		
		50	E9 00115	BLBC	R0, 14\$		
01	A3	04	90 00118	MOVB	#4, 1(R3)	0446	
		A3	7C 0011C	CLRQ	4(R3)	0452	
		76	11 0011F	BRB	19\$	0347	
		01	DD 00121	PUSHL	#1	0455	
		C7	A6 9F 00123	PUSHAB	DBG\$CS_MODE		
		52	DD 00126	PUSHL	R2		
		03	FB 00128	CALLS	#3, DBGSNMATCH		
		50	D1 0012B	CMPL	R0, #1		
		06	12 0012E	BNEQ	13\$		
01	A3	05	90 00130	MOVB	#5, 1(R3)	0457	
		61	11 00134	BRB	19\$	0347	
		04	DD 00136	PUSHL	#4	0460	
		CC	A6 9F 00138	PUSHAB	DBG\$CS_MODULE		
		52	DD 0013B	PUSHL	R2		
		03	FB 0013D	CALLS	#3, DBGSNMATCH		
		50	D1 00140	CMPL	R0, #1		
		54	12 00143	BNEQ	20\$		
		01	DD 00145	PUSHL	#1		
		0044	8F BB 00147	PUSHR	#^M<R2,R6>	0465	
		03	FB 0014B	CALLS	#3, DBGSNMATCH		
		50	E9 0014E	BLBC	R0, 16\$		
		01	DD 00151	PUSHL	#1		
		06	A6 9F 00153	PUSHAB	DBG\$CS_ALL	0471	
		52	DD 00156	PUSHL	R2		
		03	FB 00158	CALLS	#3, DBGSNMATCH		
		50	E9 0015B	BLBC	R0, 21\$		
01	A3	07	90 0015E	MOVB	#7, 1(R3)	0483	
		7D	11 00162	BRB	24\$	0465	
		0C	AC DD 00164	PUSHL	MESSAGE_VECT	0504	
		14	BB 00167	PUSHR	#^M<R2,R4>	0503	
		03	FB 00169	CALLS	#3, DBGSNSAVE_STRING		
69	03	50	E8 0016C	BLBS	R0, 17\$		
		018F	31 0016F	BRW	42\$		
		01	DD 00172	PUSHL	#1		
		0A	A6 9F 00174	PUSHAB	DBG\$CS_COMMA	0511	
		52	DD 00177	PUSHL	R2		
67		03	FB 00179	CALLS	#3, DBGSNMATCH		

11		50	E9 0017C	BLBC	R0, 18\$	
55	08	A4 9E 0017F	MOVAB	8(R4), LINK	0518	
		04 DD 00183	PUSHL	#4	0519	
68		01 FB 00185	CALLS	#1, DBG\$GET TEMPMEM		
54		50 DD 00188	MCVL	R0, NOUN NODE		
65		54 DD 0018B	MOVL	NOUN_NODE, (LINK)	0520	
		D4 11 0018E	BRB	16\$	0498	
01	A3	08 A4 D4 00190	18\$:	CLRL	8(NOUN NODE)	0527
		06 90 00193	MOV8	#6 1(R3)	0529	
		48 11 00197	19\$:	BRB	24\$	0347
		01 DD 00199	20\$:	PUSHL	#1	0535
		D3 A6 9F 0019B	PUSHAB	DBG\$CS_RADIX		
		52 DD 0019E	PUSHL	R2		
67		03 FB 001A0	CALLS	#3, DBG\$NMATCH		
01		50 D1 001A3	CMPL	R0, #1		
01	A3	26 12 001A6	BNEQ	23\$		
		14 90 001A8	MOV8	#20, 1(R3)	0540	
		01 DD 001AC	PUSHL	#1	0544	
67		8F BB 001AE	PUSHR	#^M<R2,R6>		
29		03 FB 001B2	CALLS	#3, DBG\$NMATCH		
		50 E9 001B5	BLBC	R0, 24\$		
		01 DD 001B8	PUSHL	#1	0550	
		D3 A6 9F 001BA	PUSHAB	DBG\$CS_OVERRIDE		
		52 DD 001BD	PUSHL	R2		
67		03 FB 001BF	CALLS	#3, DBG\$NMATCH		
03		50 E8 001C2	BLBS	R0, 22\$		
01	A3	010B 31 001C5	21\$:	BRW	38\$	
		15 90 001C8	MOV8	#21, 1(R3)	0562	
		13 11 001CC	BRB	24\$	0347	
		01 DD 001CE	22\$:	PUSHL	#1	0566
		D9 A6 9F 001D0	PUSHAB	DBG\$CS_SCOPE		
		52 DD 001D3	PUSHL	R2		
67		03 FB 001D5	CALLS	#3, DBG\$NMATCH		
01		50 D1 001D8	CMPL	R0, #1		
01	A3	06 12 001DB	BNEQ	25\$		
		08 90 001DD	MOV8	#8 1(R3)	0568	
		60 11 001E1	BRB	30\$	0347	
		02 DD 001E3	24\$:	PUSHL	#2	0571
		DF A6 9F 001E5	PUSHAB	DBG\$CS_SOURCE		
		52 DD 001E8	PUSHL	R2		
67		03 FB 001EA	CALLS	#3, DBG\$NMATCH		
01		50 D1 001ED	CMPL	R0, #1		
01	A3	60 12 001FO	BNEQ	31\$		
		10 90 001F2	MOV8	#16, 1(R3)	0573	
		01 DD 001F6	PUSHL	#1	0577	
		0044 8F BB 001F8	PUSHR	#^M<R2,R6>		
67		03 FB 001FC	CALLS	#3, DBG\$NMATCH		
4C		50 E9 001FF	BLBC	R0, 29\$		
		04 DD 00202	PUSHL	#4	0587	
		15 A6 9F 00204	PUSHAB	DBG\$CS_MODULE		
		52 DD 00207	PUSHL	R2		
67		03 FB 00209	CALLS	#3, DBG\$NMATCH		
0A		50 E9 0020C	BLBC	R0, 26\$		
		01 DD 0C20F	PUSHL	#1	0597	
		FE A6 9F 00211	PUSHAB	DBG\$CS_EQUAL		
		52 DD 00214	PUSHL	R2		
67		03 FB 00216	CALLS	#3, DBG\$NMATCH		

03		50	E8	00219	26\$:	BLBS	R0	27\$	
		00CC	31	0021C	27\$:	BRW	40\$		
	04	AC	DD	0021F	27\$:	PUSHL	MESSAGE VECT		0608
		AE	9F	00222		PUSHAB	MODNAMEPTR		0607
		52	DD	00225		PUSHL	R2		
69		03	FB	00227		CALLS	#3. DBG\$NSAVE_STRING		
03		50	E8	0022A		BLBS	R0	28\$	
		00D1	31	0022D	28\$:	BRW	42\$		
00000000G	00	6E	DD	00230	28\$:	PUSHL	MODNAMEPTR		0615
	64	01	FB	00232		CALLS	#1. DBG\$STA_GETSOURCEMOD		
		50	DD	00239		MOVL	R0 (NOUN_NODE)		
		55	12	0023C		BNEQ	33\$		0620
		6E	DD	0023E		PUSHL	MODNAMEPTR		0624
		01	DD	00240		PUSHL	#1		0623
6A	000281E8	8F	DD	00242		PUSHL	#164328		
		03	FB	00248		CALLS	#3. DBG\$NMAKE_ARG_VECT		
		00AF	31	0024B		BRW	41\$		
		64	D4	0024E	29\$:	CLRL	(NOUN_NODE)		0632
		7F	11	00250	30\$:	BRB	37\$		0347
		01	DD	00252	31\$:	PUSHL	#1		0636
	E6	A6	9F	00254		PUSHAB	DBGSCS_TRACE		
		52	DD	00257		PUSHL	R2		
67		03	FB	00259		CALLS	#3. DBG\$NMATCH		
01		50	D1	0025C		CMPL	R0 #1		
		06	12	0025F		BNEQ	32\$		
01	A3	09	90	00261		MOVB	#9. 1(R3)		0638
		41	11	00265		BRB	35\$		0641
		02	DD	00267	32\$:	PUSHL	#2		0645
	EC	A6	9F	00269		PUSHAB	DBGSCS_TYPE		
		52	DD	0026C		PUSHL	R2		
67		03	FB	0026E		CALLS	#3. DBG\$NMATCH		
01		50	D1	00271		CMPL	R0 #1		
		1F	12	00274		BNEQ	34\$		
		01	DD	00276		PUSHL	#1		0652
67	0044	8F	BB	00279		PUSHR	#^M<R2,R6>		
51		03	FB	0027C		CALLS	#3. DBG\$NMATCH		
		50	E9	0027F		BLBC	R0, 38\$		
		01	DD	00282		PUSHL	#1		0667
	1C	A6	9F	00284		PUSHAB	DBGSCS_OVERRIDE		
		52	DD	00287		PUSHL	R2		
67		03	FB	00289		CALLS	#3. DBG\$NMATCH		
44		50	E9	0028C		BLBC	R0, 38\$		
01	A3	0D	90	0028F		MOVB	#1\$. 1(R3)		0679
		70	11	00293	33\$:	BRB	43\$		0347
		01	DD	00295	34\$:	PUSHL	#1		0682
	F1	A6	9F	00297		PUSHAB	DBGSCS_WATCH		
		52	DD	0029A		PUSHL	R2		
67		03	FB	0029C		CALLS	#3. DBG\$NMATL		
01		50	D1	0029F		CMPL	R0 #1		
		11	12	002A2		BNEQ	36\$		
01	A3	0E	90	002A4		MOVB	#14. 1(R3)		0684
	OC	AC	DD	002A8	35\$:	PUSHL	MESSAGE VECT		0687
		OC	BB	002AB		PUSHR	#^M<R2,R3>		0685
00000000G	00	03	FB	002AD		CALLS	#3. DBG\$EVENT_SHOW_CANCEL_SYNTAX		
		04	002B4			RET			
		03	DD	002B5	36\$:	PUSHL	#3		0693
	F7	A6	9F	002B7		PUSHAB	DBGSCS_WINDOW		

		52	DD 002BA	PUSHL	R2	
		03	FB 002BC	CALLS	#3, DBG\$NMATCH	
		50	D1 002BF	CMPL	R0, #1	
		0F	12 002C2	BNEQ	38\$	
01	A3	13	90 002C4	MOVB	#19, 1(R3)	0605
		0C	BB 002C8	PUSHR	#^M<R2,R3>	0696
00000000G	00	02	FB 002CA	CALLS	#2, DBG\$SCR_PARSE_CANWIND_CMD	0347
		32	11 002D1	37\$:	BRB 43\$	0706
		01	DD 002D3	38\$:	PUSHL #1	
		04	A6 9F 002D5	PUSHAB	DBG\$CS_CR	
		52	DD 002D9	PUSHL	R2	
67		03	FB 002DA	CALLS	#3, DBG\$NMATCH	
08		50	E9 002DD	BLBC	R0, 40\$	
	000280D0	8F	DD 002E0	PUSHL	#164048	0708
6A		01	FB 002E6	39\$:	CALLS #1, DBG\$NMAKE_ARG_VECT	
		12	11 002E9	BRB 41\$		
00000000G	00	52	DD 002EB	40\$:	PUSHL R2	0710
		01	FB 002ED	CALLS	#1, DBG\$NNEXT_WORD	
00000000G	00	50	DD 002F4	PUSHL	R0	
OC	BC	01	FB 002F6	CALLS	#1, DBG\$NSYNTAX_ERROR	
		50	DO 002FD	41\$:	MOVL R0, @MESSAGE_VECT	0705
		04	DO 00301	42\$:	MOVL #4, R0	0712
			04 00304	RET		
		50	01 DO 00305	43\$:	MOVL #1, R0	0717
			04 00308	RET		0719

: Routine Size: 777 bytes.    Routine Base: DBG\$CODE + 0000

```
590      0720 1 GLOBAL ROUTINE DBGS$NEXECUTE_CANCEL (VERB_NODE, MESSAGE_VECT) =
591      0721 1
592      0722 1 !++
593      0723 1 FUNCTIONAL DESCRIPTION:
594      0724 1
595      0725 1 This routine uses the command execution tree constructed by the parse
596      0726 1 network as input and performs the semantic actions associated with
597      0727 1 the given input corresponding to the CANCEL xxx command. If the command
598      0728 1 cannot be executed, a message argument vector is constructed and returned.
599      0729 1
600      0730 1 FORMAL PARAMETERS:
601      0731 1
602      0732 1     VERB_NODE      - A longword containing the address of the head node
603      0733 1                  of the command execution tree. This corresponds to
604      0734 1                  the verb node.
605      0735 1
606      0736 1     MESSAGE_VECT   - The address of a longword to contain the address of
607      0737 1                  a standard message argument vector upon detection of
608      0738 1                  errors.
609      0739 1
610      0740 1 IMPLICIT INPUTS:
611      0741 1
612      0742 1     The linked list command execution tree pointed to by verb_node.
613      0743 1
614      0744 1 IMPLICIT OUTPUTS:
615      0745 1
616      0746 1     On failure, a message argument vector is constructed and returned.
617      0747 1
618      0748 1 ROUTINE VALUE:
619      0749 1
620      0750 1     An unsigned integer longword completion code
621      0751 1
622      0752 1 COMPLETION CODES:
623      0753 1
624      0754 1     STSSK_SUCCESS (1)      - Success. Command executed.
625      0755 1
626      0756 1     STSSK_SEVERE (4)    - Failure. Command not executed. Message argument
627      0757 1                  vector constructed and returned.
628      0758 1
629      0759 1 SIDE EFFECTS:
630      0760 1
631      0761 1     Various semantic actions corresponding to the CANCEL xxx command are
632      0762 1                  performed.
633      0763 1
634      0764 1 !-- BEGIN
635      0765 2
636      0766 2
637      0767 2 MAP
638      0768 2     VERB_NODE: REF DBGS$VERB_NODE; ! Pointer to command Verb Node
639      0769 2
640      0770 2 LOCAL
641      0771 2     NOUN_NODE: REF DBGS$NOUN_NODE; ! Pointer to a command Noun Node
642      0772 2     ADDR_EXP_DESC; ! Address expression descriptor
643      0773 2     ADDRESS_VECTOR [2]; ! Address and bit offset
644      0774 2     TYPE; ! Type of AED described object
645      0775 2
646      0776 2
```

```
: 647      0777 2
: 648      0778 2  ! Recover the noun node
: 649      0779 2
: 650      0780 2 NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
: 651      0781 2
: 652      0782 2
: 653      0783 2
: 654      0784 2  ! Perform the indicated action base on the verb composite
: 655      0785 2 CASE .VERB_NODE[DBG$B_VERB_COMPOSITE] FROM CANCEL_MINIMUM TO CANCEL_MAXIMUM OF
: 656      0786 2   SET
: 657      0787 2
: 658      0788 2
: 659      0789 2  ! Execute the CANCEL ALL command.
: 660      0790 2
: 661      0791 2 [CANCEL_ALL]:
: 662      0792 2   BEGIN
: 663      0793 2     LOCAL
: 664      0794 2       SCOPE_LIST,
: 665      0795 2       DUMMY;
: 666      0796 2
: 667      0797 2  ! Just cancel everything in sight
: 668      0798 2
: 669      0799 2 scope_list = 0;
: 670      0800 2 dbg$gl_context [dbg$k_all] = true;
: 671      0801 2
: 672      0802 2 DBGSEVENT_CANCEL_ALL ();
: 673      0803 2
: 674      0804 2
: 675      0805 2   dbg$runframe [dbg$v_trace_all] = false;    ! For next two calls
: 676      0806 2   dbg$gb_resignal = true;           ! Exception break
: 677      0807 2   dbg$set_mod_def ();           ! Set mode defaults
: 678      0808 2   dbg$set_stp_def ();           ! Set step defaults
: 679      0809 2   dbg$rst_setscope (scope_list, dummy); ! Scopes (new debugger)
: 680      0810 2   dbg$gl_gbltyp = -1;            ! Override type
: 681      0811 2   dbg$gw_gbllength = 0;          ! Override length
: 682      0812 2   dbg$gl_dfltyp = dsc$k_dtype_l; ! Default type
: 683      0813 2   dbg$gw_dfltlen = 4;           ! Default length
: 684      0814 2 END;
: 685      0815 2
: 686      0816 2 [cancel_break]: ! CANCEL BREAK <ADDR_EXP>
: 687      0817 2   RETURN DBGSEVENT_SHOW_CANCEL_SEMATICS (.VERB_NODE,
: 688      0818 2           .MESSAGE_VECT
: 689      0819 2 );
: 690      0820 2
: 691      0821 2  ! Execute the CANCEL DEVELOPER 0, 1, ..., n command. Cancel all bits
: 692      0822 2  ! in DBG$GL_DEVELOPER indicated on the command. If no bits are speci-
: 693      0823 2  ! fied, clear all developer bits.
: 694      0824 2
: 695      0825 2 [CANCEL_DEVELOPER]:
: 696      0826 2   BEGIN
: 697      0827 2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
: 698      0828 2     IF .NOUN_NODE EQ 0 THEN DBG$GL_DEVELOPER = 0;
: 699      0829 3     WHILE .NOUN_NODE NEQ 0 DO
: 700      0830 4       BEGIN
: 701      0831 4         DBG$GL_DEVELOPER [.NOUN_NODE[DBG$L_NOUN_VALUE]] = FALSE;
: 702      0832 4         NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
: 703      0833 3     END;
```

```
704 0834 3
705 0835 2
706 0836 2
707 0837 2
708 0838 2
709 0839 2
710 0840 2
711 0841 2
712 0842 2
713 0843 2
714 0844 2
715 0845 2
716 0846 2
717 0847 3
718 0848 3
719 0849 3
720 0850 2
721 0851 2
722 0852 2
723 0853 2
724 0854 2
725 0855 2
726 0856 3
727 0857 3
728 0858 3
729 0859 3
730 0860 3
731 0861 2
732 0862 2
733 0863 2
734 0864 3
735 0865 3
736 0866 3
737 0867 3
738 0868 3
739 0869 3
740 0870 3
741 0871 3
742 0872 3
743 0873 4
744 0874 4
745 0875 4
746 0876 4
747 0877 4
748 0878 4
749 0879 4
750 0880 5
751 0881 5
752 0882 5
753 0883 5
754 0884 5
755 0885 4
756 0886 4
757 0887 4
758 0888 4
759 0889 4
760 0890 4

    END;

    ! Execute the CANCEL DISPLAY command.

[CANCEL_DISPLAY]:
    DBG$SCR_EXECUTE_CANDISP_CMD(.VERB_NODE);

    ! Execute the CANCEL EXCEPTION BREAK command.

[CANCEL_EXCEPTION_BREAK]:
    BEGIN
        DBG$GB_RESIGNAL = TRUE;
        RETURN DBG$EVENT_SHOW_CANCEL_SEMANTICS( VFB_NODE..MESSAGE_VECT);
    END;

    ! Execute the CANCEL MODE command.

[CANCEL_MODE]:
    BEGIN
        DBG$gb_radix[dbg$gb_radix_input] = dbg$ngt_trans_radix(dbg$k_default);
        DBG$gb_radix[dbg$gb_radix_output] = dbg$ngt_trans_radix(dbg$k_default);
        DBG$gb_radix[dbg$gb_radix_output_over] = dbg$k_default;
        DBG$SET_MOD_DEF();
    END;

[cancel_module] :      ' CANCEL MODULE or CANCEL MODULE/ALL
    BEGIN
        ! Module names are stored away as counted strings
        !
        LOCAL
            NAME_BUFF : REF VECTOR [,BYTE]; ! Module name buffer
        WHILE .noun_node NEQA 0
        DO
            BEGIN
                ! Retrieve the name buffer and call the symbol table
                !
                name_buff = .noun_node [dbg$l_noun_value];
                IF NOT dbg$rst_cnamod (name_buff [T], .name_buff [0])
                THEN
                    BEGIN
                        .message_vect = dbg$nmake_arg_vect (dbg$nosuchmodu,
                            1,
                            name_buff [0]);
                        RETURN sts$k_severe;
                    END;
                !
                ! Obtain the next noun node
                !
                noun_node = .noun_node [dbg$l_noun_link];
            END;
    END;
```

761 0891 4  
762 0892 3  
763 0893 3  
764 0894 2  
765 0895 2  
766 0896 2  
767 0897 3  
768 0898 3  
769 0899 2  
770 0900 2  
771 0901 2  
772 0902 3  
773 0903 3  
774 0904 3  
775 0905 3  
776 0906 2  
777 0907 2  
778 0908 2  
779 0909 2  
780 0910 2  
781 0911 2  
782 0912 3  
783 0913 3  
784 0914 3  
785 0915 3  
786 0916 3  
787 0917 3  
788 0918 3  
789 0919 2  
790 0920 2  
791 0921 2  
792 0922 3  
793 0923 3  
794 0924 2  
795 0925 2  
796 0926 2  
797 0927 2  
798 0928 2  
799 0929 2  
800 0930 2  
801 0931 2  
802 0932 2  
803 0933 2  
804 0934 3  
805 0935 3  
806 0936 3  
807 0937 2  
808 0938 2  
809 0939 2  
810 0940 2  
811 0941 2  
812 0942 2  
813 0943 2  
814 0944 2  
815 0945 2  
816 0946 3  
817 0947 2

END; . End of Loop  
END;  
[cancel\_module\_all] :  
BEGIN  
dbg\$rst\_canmod (0, 0);  
END;  
  
[cancel\_radix] :  
BEGIN  
dbg\$gb\_radix[dbg\$b\_radix\_input] = dbg\$ngt\_trans\_radix(dbg\$k\_default);  
dbg\$gb\_radix[dbg\$b\_radix\_output] = dbg\$ngt\_trans\_radix(dbg\$k\_default);  
dbg\$gb\_radix[dbg\$b\_radix\_output\_over] = dbg\$k\_default;  
END;  
  
[cancel\_radix\_override]:  
dbg\$gb\_radix[dbg\$b\_radix\_output\_over] = dbg\$k\_default;  
  
[cancel\_scope] :  
BEGIN  
LOCAL  
DUMMY,  
SCOPE\_LIST;  
  
scope\_list = 0;  
dbg\$rst\_setscope (scope\_list, dummy);  
END;  
  
[cancel\_source] : ! CANCEL SOURCE[/MODULE=modname]  
BEGIN  
dbg\$src\_cancel\_source(.noun\_node[dbg\$1\_noun\_value]);  
END;  
  
[cancel\_trace] : ! CANCEL TRACE <ADDR\_EXP>  
RETURN DBGSEVENT\_SHOW\_CANCEL\_SEMATICS (.VERB\_NODE,  
.MESSAGE\_VECT  
);  
  
! Execute the CANCEL TYPE/OVERRIDE command.  
[CANCEL\_TYPE\_OVERRIDE]:  
BEGIN  
DBG\$GL\_GBLTYP = -1;  
DBG\$GW\_GBLNGTH = 0;  
END;  
  
! Execute the CANCEL WATCH <addr-expr> command.  
[CANCEL\_WATCH]:  
RETURN DBGSEVENT\_SHOW\_CANCEL\_SEMATICS (.VERB\_NODE,  
.MESSAGE\_VECT  
);  
! Execute the CANCEL WINDOW command.

```
818      0948 2      [CANCEL_WINDOW]:  
819      0949 2      DBG$SCR_EXECUTE_CANWIND_CMD(.VERB_NODE);  
820      0950 2  
821      0951 2  
822      0952 2      ; Any other CASE index constitutes and internal DEBUG error.  
823      0953 2  
824      0954 2      [INRANGE,OUTRANGE]:  
825      0955 2      $DBG_ERROR('DBGNCANCL\NEXECUTE_CANCEL');  
826      0956 2  
827      0957 2      TES;  
828      0958 2  
829      0959 2      RETURN STSSUCCESS;  
830      0960 2  
831      0961 1      END;
```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0  
45 58 45 4E 5C 4C 43 4E 41 43 4E 47 42 44 19 00084 P.AAX: .ASCII <25>\DBGNCANCL\<92>\NEXECUTE\_CANCEL\

00000000G	00	00000000'	EF	9F 0006D	2\$:	PUSHAB	24\$-1\$,-		0955
		00028362	01	DD 00073		PUSHL	16\$-1\$,-		
			03	FB 00075		PUSHL	17\$-1\$		
			69	11 00082		CALLS	P.AAX		
			AE	D4 00084	3\$:	BRB	#1		
00000000G	00		04	88 00087		CLRL	#164706		0799
00000000G	00		02	FB 0008E		BISB2	#3, LIB\$SIGNAL		0800
00000000G	00		00	8A 00095		CALLS	8\$		0802
00000000G	67		04	90 0009C		BICB2	SCOPE_LIST		0804
00000000G	68		01	FB 0009F		MOVB	#2, DBG\$GL_CONTEXT+1		0805
00000000G	00		00	FB 000A2		CALLS	#0, DBG\$EVENT_CANCEL_ALL		0806
			5E	DD 000A9		CALLS	#4, DBG\$RUNFRAME+72		0807
			AE	9F 000AB		PUSHL	#1, DBG\$GB_RESIGNAL		0808
			69	02 FB 000AE		CALLS	#0, DBG\$SET_MOD_DEF		
			6A	01 CE 000B1		MNEGL	#0, DBG\$SET_STP_DEF		
00000000G	00		68	84 000B4		CLRW	SP		
00000000G	00		08	D0 000B6		MOVL	SCOPE_LIST		0809
00000000G	00		04	B0 000BD		MOVW	#2, DBG\$RST_SETSCOPE		0810
			47	11 000C4		BRB	#0, DBG\$GL_GBLTYP		0811
			53	A2 D0 000C6	4\$:	MOVL	DBGSGL_GBLNGTH		0812
			06	12 000CA		BNEQ	#8, DBG\$GL_DFLTTYP		0785
00000000G	00	00000000G	00	D4 000CC		CLRL	#4, DBG\$GW_DFLTLENG		0827
			53	D5 000D2	5\$:	TSTL	11\$		0828
			78	13 000D4		BEQL	8(R2), NOUN_NODE		0829
00000000G	00		63	E5 000D6		BBCC	5\$		
00000000G	53	08	A3	D0 000DE	6\$:	MOVL	DBG\$GL_DEVELOPER		0831
			EE	11 000E2		BRB	NOUN_NODE		0832
00000000G	00		52	DD 000E4	7\$:	PUSHL	15\$		0829
			01	FB 000E6		CALLS	(NOUN_NODE), DBG\$GL_DEVELOPER, 6\$		0841
			76	11 000ED	8\$:	BRB	8(NOUN_NODE), NOUN_NODE		
			01	90 000EF	9\$:	MOVW	5\$		
00000000G	00		0092	31 000F2		BRW	R2		
			01	DD 000F5	10\$:	PUSHL	#1, DBGS\$SCR_EXECUTE_CANDISP_CMD		
			66	01 FB 000F7		CALLS	#1, DBGS\$GET_TRANS_RADIX		0848
			65	50 90 000FA		MOVB	R0, DBGS\$GB_RADIX		0849
			01	DD 000FD		PUSHL	#1		0857
01	66		01	FB 000FF		CALLS	#1, DBGS\$GET_TRANS_RADIX		
02	A5		50	90 00102		MOVB	R0, DBGS\$GB_RADIX+1		0859
	A5		01	90 00106		MOVB	#1, DBGS\$GB_RADIX+2		0860
	68		00	FB 0010A		CALLS	#0, DBG\$SET_MOD_DEF		0785
			76	11 0010D	11\$:	BRB	22\$		0871
			53	D5 0010F	12\$:	TSTL	NOUN_NODE		
			72	13 00111		BEQL	22\$		
00000000G	54		63	D0 00113		MOVL	(NOUN_NODE), NAME_BUFF		0877
00000000G	7E		64	9A 00116		MOVZBL	(NAME_BUFF) -(SPT)		0878
	01		A4	9F 00119		PUSHAB	1(NAME_BUFF)		
00000000G	00		02	FB 0011C		CALLS	#2, DBG\$RST_CANMOD		
	19		50	E8 00123		BLBS	R0, 13\$		
			54	DD 00126		PUSHL	NAME_BUFF		
			01	DD 00128		PUSHL	#1		
00000000G	00	000281E8	8F	DD 0012A		PUSHL	#164328		
	08		03	FB 00130		CALLS	#3, DBGS\$MAKE_ARG_VECT		
	BC		50	DD 00137		MOVL	R0, MESSAGE_VECT		0883

	50	04	00 0013B	MOVL #4, R0	: 0884
		04	0013E	RET	
	53	08	A3 D0 0013F	13\$: MOVL 8(NOUN_NODE), NOUN_NODE	: 0890
		CA 11	00143	BRB 12\$	: 0871
00000000G	00	7E	7C 00145	14\$: CLRQ -(SP)	: 0898
		02	FB 00147	CALLS #2 DBG\$RST_CANMOD	: 0785
		4D	11 0014E	15\$: BRB 25\$	: 0903
		01	DD 00150	16\$: PUSHL #1	
	66	01	FB 00152	CALLS #1, DBG\$NGET_TRANS_RADIX	
	65	50	90 00155	MOV8 R0, DBG\$GB_RADIX	
		01	DD 00158	PUSHL #1	: 0904
01	66	01	FB 0015A	CALLS #1, DBG\$NGET_TRANS_RADIX	
02	A5	50	90 0015D	MOV8 R0, DBG\$GB_RADIX+1	
		01	90 00161	17\$: MOV8 #1, DBG\$GB_RADIX+2	: 0909
		36	11 00165	18\$: BRB 25\$	
		0C	AE D4 00167	19\$: CLRL SCOPE_LIST	: 0917
		08	AE 9F 0016A	PUSHAB DUMMY	: 0918
		10	AE 9F 0016D	PUSHAB SCOPE_LIST	
	69	02	FB 00170	CALLS #2 DBG\$RST_SETSCOPE	
		28	11 00173	BRB 25\$	: 0785
00000000G	00	63	DD 00175	20\$: PUSHL (NOUN_NODE)	: 0923
		01	FB 00177	CALLS #1, DBG\$SRC_CANCEL_SOURCE	
		1D	11 0017E	BRB 25\$	: 0785
	6A	01	CE 00180	21\$: MNEGL #1, DBG\$GL_GBLTYP	: 0935
		68	B4 00183	CLRW DBG\$GW_GBL[NGTH]	: 0936
		16	11 00185	22\$: BRB 25\$	: 0785
		08	AC DD 00187	23\$: PUSHL MESSAGE_VECT	: 0944
00000000G	00	52	DD 0018A	PUSHL R2	: 0943
		02	FB 0018C	CALLS #2, DBG\$EVENT_SHOW_CANCEL_SEMATICS	
		04	00193	RET	
00000000G	00	52	DD 00194	24\$: PUSHL R2	: 0949
		01	FB 00196	CALLS #1, DBG\$SCR_EXECUTE_CANWIND_CMD	
	50	01	DD 0019D	25\$: MOVL #1, R0	: 0959
		04	001A0	RET	: 0961

: Routine Size: 417 bytes, Routine Base: DBG\$CODE + 0309

: 832 0962 1  
: 833 0963 0 END ELUDOM

.EXTRN LIB\$SIGNAL

## PSECT SUMMARY

Name	Bytes	Attributes
DBG\$PLIT	158 NOVEC,NOWRT,	RD : EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$CODE	1194 NOVEC,NOWRT,	RD : EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)

## Library Statistics

File	Total	Symbols	Pages	Processing		
		Total	Loaded	Percent	Mapped	Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	3	0	1000	00:01.8	
-\$255\$DUA28:[DEBUG.OBJ]STRU[DEF.L32;1	32	0	0	7	00:00.1	
-\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	28	1	97	00:02.0	
-\$255\$DUA28:[DEBUG.OBJ]DSTREC(RDS.L32;1	418	0	0	31	00:00.3	
-\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	6	1	22	00:00.3	
-\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	0	0	12	00:00.3	

## COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGNCANCL/OBJ=OBJ\$:DBGNCANCL MSRC\$:DBGNCANCL/UPDATE=(ENH\$:DBGNCANCL)

: Size. 1194 code + 158 data bytes  
: Run Time: 00:27.3  
: Elapsed Time: 01:28.7  
: Lines/CPU Min: 2119  
: Lexemes/CPU-Min: 9625  
: Memory Used: 320 pages  
: Compilation Complete

0086 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

DBGMOD  
LIS

DBGMSG  
LIS

DBGNCANCEL  
LIS

DBGNCNTR  
LIS